Network Working Group
Request for Comments: 1561
Category: Experimental
D. Piscitello
Core Competence
December 1993

Use of ISO CLNP in TUBA Environments

Status of this Memo

Abstract

This memo specifies a profile of the ISO/IEC 8473 Connectionless-mode Network Layer Protocol (CLNP, [1]) for use in conjunction with RFC 1347, TCP/UDP over Bigger Addresses (TUBA, [2]). It describes the use of CLNP to provide the lower-level service expected by Transmission Control Protocol (TCP, [3]) and User Datagram Protocol (UDP, [4]). CLNP provides essentially the same datagram service as Internet Protocol (IP, [5]), but offers a means of conveying bigger network addresses (with additional structure, to aid routing).

While the protocols offer nearly the same services, IP and CLNP are not identical. This document describes a means of preserving the semantics of IP information that is absent from CLNP while preserving consistency between the use of CLNP in Internet and OSI environments. This maximizes the use of already-deployed CLNP implementations.

Acknowledgments

Many thanks to Ross Callon (Wellfleet Communications), John Curran (BBN), Cyndi Jung (3Com), Paul Brooks (UNSW), Brian Carpenter (CERN), Keith Sklower (Cal Berkeley), Dino Farinacci and Dave Katz (Cisco Systems), Rich Colella (NIST/CSL) and David Oran (DEC) for their assistance in composing this text.

Conventions

The following language conventions are used in the items of specification in this document:

* MUST, SHALL, or MANDATORY -- the item is an absolute requirement of the specification.

* SHOULD or RECOMMENDED -- the item should generally be followed for all but exceptional circumstances.

* MAY or OPTIONAL -- the item is truly optional and may be followed or ignored according to the needs of the implementor.

1. Terminology

To the extent possible, this document is written in the language of the Internet. For example, packet is used rather than "protocol data unit", and "fragment" is used rather than "segment". There are some terms that carry over from OSI; these are, for the most part, used so that cross-reference between this document and RFC 994 [6] or ISO/IEC 8473 is not entirely painful. OSI acronyms are for the most part avoided.

2. Introduction

The goal of this specification is to allow compatible and interoperable implementations to encapsulate TCP and UDP packets in CLNP data units. In a sense, it is more of a "hosts requirements" document for the network layer of TUBA implementations than a protocol specification. It is assumed that readers are familiar with STD 5, RFC 791, STD 5, RFC 792 [7], STD 3, RFC 1122 [8], and, to a lesser extent, RFC 994 and ISO/IEC 8473. This document is compatible with (although more restrictive than) ISO/IEC 8473; specifically, the order, semantics, and processing of CLNP header fields is consistent between this and ISO/IEC 8473.

[Note: RFC 994 contains the Draft International Standard version of ISO CLNP, in ASCII text. This is not the final version of the ISO/IEC protocol specification; however, it should provide sufficient background for the purpose of understanding the relationship of CLNP to IP, and the means whereby IP information is to be encoded in CLNP header fields. Postscript versions of ISO CLNP and associated routing protocols are available via anonymous FTP from merit.edu, and may be found in the directory /pub/ISO/IEC.

3. Overview of CLNP

ISO CLNP is a datagram network protocol. It provides fundamentally the same underlying service to a transport layer as IP. CLNP provides essentially the same maximum datagram size, and for those circumstances where datagrams may need to traverse a network whose maximum packet size is smaller than the size of the datagram, CLNP provides mechanisms for fragmentation (data unit identification, fragment/total length and offset). Like IP, a checksum computed on the CLNP header provides a verification that the information used in processing the CLNP datagram has been transmitted correctly, and a lifetime control mechanism ("Time to Live") imposes a limit on the amount of time a datagram is allowed to remain in the internet system. As is the case in IP, a set of options provides control functions needed or useful in some situations but unnecessary for the most common communications.

Note that the encoding of options differs between the two protocols, as do the means of higher level protocol identification. Note also that CLNP and IP differ in the way header and fragment lengths are represented, and that the granularity of lifetime control (time-to-live) is finer in CLNP.

Some of these differences are not considered "issues", as CLNP provides flexibility in the way that certain options may be specified and encoded (this will facilitate the use and encoding of certain IP options without change in syntax); others, e.g., higher level protocol identification and timestamp, must be accommodated in a transparent manner in this profile for correct operation of TCP and UDP, and continued interoperability with OSI implementations. Section 4 describes how header fields of CLNP must be populated to satisfy the needs of TCP and UDP.

Errors detected during the processing of a CLNP datagram MAY be reported using CLNP Error Reports. Implementations of CLNP for TUBA environments MUST be capable of processing Error Reports (this is consistent with the 1992 edition (2) of the ISO/IEC 8473 standard). Control messages (e.g., echo request/reply and redirect) are similarly handled in CLNP, i.e., identified as separate network layer packet types. The relationship between CLNP Error and Control messages and Internet Control Message Protocol (ICMP, [7]), and issues relating to the handling of these messages is described in Section 5.

Table 1 provides a high-level comparison of CLNP to IP:

Function	ISO CLNP	DOD IP
Header Length	indicated in octets	in 32-bit words
Version Identifier	1 octet	4 bits
Lifetime (TTL)	500 msec units	1 sec units
Flags	Fragmentation allowed,	Don’t Fragment,
	More Fragments	More Fragments,
	Suppress Error Reports	<not defined>
Packet Type	5 bits	<not defined>
Fragment Length	16 bits, in octets	16 bits, in octets
Header Checksum	16-bit (Fletcher)	16-bit
Total Length	16 bits, in octets	<not defined>
Addressing	Variable length	32-bit fixed
Data Unit Identifier	16 bits	16 bits
Fragment offset	16 bits, in octets	13 bits, 8-octet units
Higher Layer Protocol	Selector in address	Protocol
Options	Security	Security
	Priority	TOS Precedence bits
	Complete Source Route	Strict Source Route
	Quality of Service	Type of Service
	Partial Source Route	Loose Source Route
	Record Route	Record Route
	Padding	Padding
	<defined herein>	Timestamp

Table 1. Comparison of IP to CLNP

The composition and processing of a TCP pseudo-header when CLNP is used to provide the lower-level service expected by TCP and UDP is described in Section 6.

[Note: This experimental RFC does not discuss multicasting. Presently, there are proposals for multicast extensions for CLNP in ISO/IEC/JTC1/SC6, and a parallel effort within TUBA. A future revision to this RFC will incorporate any extensions to CLNP that may be introduced as a result of the adoption of one of these alternatives.]

4. Proposed Internet Header using CLNP

A summary of the contents of the CLNP header, as it is proposed for use in TUBA environments, is illustrated in Figure 4-1:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        ........Data Link Header........       | NLP ID        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Header Length  |    Version    | Lifetime (TTL)|Flags|  Type   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Fragment Length        |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Dest Addr Len |               Destination Address...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| PROTO field   | Src  Addr Len |  Source  Address...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Source Address |   Reserved    |       Data Unit Identifier    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Fragment Offset       |   Total Length of packet      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Options  (see Table 1)                      |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               Data                            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that each tick mark represents one bit position.

Figure 4-1. CLNP for TUBA

Note 1: For illustrative purposes, Figure 4-1 shows Destination and Source Addresses having a length of 19 octets, including the PROTO/reserved field. In general, addresses can be variable length, up to a maximum of 20 octets, including the PROTO/reserved field.

Note 2: Due to differences in link layer protocols, it is not possible to ensure that the packet starts on an even alignment. Note, however, that many link level protocols over which CLNP is operated use a odd length link (e.g., IEEE 802.2). (In Figure 4-1, the rest of the CLNP packet is even-aligned.)

The encoding of CLNP fields for use in TUBA environments is as follows.

4.1 Network Layer Protocol Identification (NLP ID)

This one-octet field identifies this as the ISO/IEC 8473 protocol; it MUST set to binary 1000 0001.

4.2 Header Length Indication (Header Length)

Header Length is the length of the CLNP header in octets, and thus points to the beginning of the data. The value 255 is reserved. The header length is the same for all fragments of the same (original) CLNP packet.

4.3 Version

This one-octet field identifies the version of the protocol; it MUST be set to a binary value 0000 0001.

4.4 Lifetime (TTL)

Like the TTL field of IP, this field indicates the maximum time the datagram is allowed to remain in the internet system. If this field contains the value zero, then the datagram MUST be destroyed; a host, however, MUST NOT send a datagram with a lifetime value of zero. This field is modified in internet header processing. The time is measured in units of 500 milliseconds, but since every module that processes a datagram MUST decrease the TTL by at least one even if it process the datagram in less than 500 millisecond, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum CLNP datagram lifetime. [Like IP, the colloquial usage of TTL in CLNP is as a coarse hop-count.]

Unless otherwise directed, a host SHOULD use a value of 255 as the initial lifetime value.

4.5 Flags

Three flags are defined. These occupy bits 0, 1, and 2 of the Flags/Type octet:

```
  0   1   2
+---+---+---+
| F | M | E |
| P | F | R |
+---+---+---+
```

The Fragmentation Permitted (FP) flag, when set to a value of one (1), is semantically equivalent to the "may fragment" value of the Don't Fragment field of IP; similarly, when set to zero (0), the Fragmentation Permitted flag is semantically equivalent to the "Don't Fragment" value of the Don't Fragment Flag of IP.

[Note: If the Fragmentation Permitted field is set to the value 0, then the Data Unit Identifier, Fragment Offset, and Total Length fields are not present. This denotes a single fragment datagram. In such datagrams, the Fragment Length field contains the total length of the datagram.]

The More Fragments flag of CLNP is semantically and syntactically the same as the More Fragments flag of IP; a value of one (1) indicates that more segments/fragments are forthcoming; a value of zero (0) indicates that the last octet of the original packet is present in this segment.

The Error Report (ER) flag is used to suppress the generation of an error message by a host/router that detects an error during the processing of a CLNP datagram; a value of one (1) indicates that the host that originated this datagram thinks error reports are useful, and would dearly love to receive one if a host/router finds it necessary to discard its datagram(s).

4.6 Type field

The type field distinguishes data CLNP packets from Error Reports from Echo packets. The following values of the type field apply:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
|   flags   | 1 | 1 | 1 | 0 | 0 |  => Encoding of Type = data packet
+---+---+---+---+---+---+---+---+
|   flags   | 0 | 0 | 0 | 0 | 1 |  => Encoding of Type = error report
+---+---+---+---+---+---+---+---+
|   flags   | 1 | 1 | 1 | 1 | 0 |  => Encoding of Type = echo request
+---+---+---+---+---+---+---+---+
|   flags   | 1 | 1 | 1 | 1 | 1 |  => Encoding of Type = echo reply
+---+---+---+---+---+---+---+---+
```

Error Report packets are described in Section 5.

Echo packets and their use are described in RFC 1139 [9].

4.7 Fragment Length

Like the Total Length of the IP header, the Fragment length field contains the length in octets of the fragment (i.e., this datagram) including both header and data.

[Note: CLNP also may also have a Total Length field, that contains the length of the original datagram; i.e., the sum of the length of the CLNP header plus the length of the data submitted by the higher level protocol, e.g., TCP or UDP. See Section 4.12.]

4.8 Checksum

A checksum is computed on the header only. It MUST be verified at each host/router that processes the packet; if header fields are changed during processing (e.g., the Lifetime), the checksum is modified. If the checksum is not used, this field MUST be coded with a value of zero (0). See Appendix A for algorithms used in the computation and adjustment of the checksum. Readers are encouraged to see [10] for a description of an efficient implementation of the checksum algorithm.

4.9 Addressing

CLNP uses OSI network service access point addresses (NSAPAs); NSAPAs serve the same identification and location functions as an IP address, plus the protocol selector value encoded in the IPv4 datagram header, and with additional hierarchy. General purpose

CLNP implementations MUST handle NSAP addresses of variable length up to 20 octets, as defined in ISO/IEC 8348 [11]. TUBA implementations, especially routers, MUST accommodate these as well. Thus, for compatibility and interoperability with OSI use of CLNP, the initial octet of the Destination Address is assumed to be an Authority and Format Indicator, as defined in ISO/IEC 8348. NSAP addresses may be between 8 and 20 octets long (inclusive).

TUBA implementations MUST support both ANSI and GOSIP style addresses; these are described in RFC 1237 [12], and illustrated in Figure 4-2. RFC 1237 describes the ANSI/GOSIP initial domain parts as well as the format and composition of the domain specific part. It is further recommended that TUBA implementations support the assignment of system identifiers for TUBA/CLNP hosts defined in [13] for the purposes of host address autoconfiguration as described in [14]. Additional considerations specific to the interpretation and encoding of the selector part are described in sections 4.9.2 and 4.9.4.

```
         +-------------+
         | <-- IDP --> |
         +----+--------+-----------------------------------+
         |AFI |  IDI   |           <-- DSP -->             |
         +----+--------+----+---+-----+----+-----+---+-----+
         | 47 |  0005  |DFI |AA |Rsvd | RD |Area |ID |Sel |
         +----+--------+----+---+-----+----+-----+---+-----+
  octets | 1  |   2    | 1  | 3 |  2  | 2  | 2   | 6 | 1  |
         +----+--------+----+---+-----+----+-----+---+-----+
```

Figure 4-2 (a): GOSIP Version 2 NSAP structure.

```
         +-------------+
         |<-- IDP -->  |
         +----+--------+-----------------------------------+
         |AFI |  IDI   |          <-- DSP -->              |
         +----+--------+----+---+-----+----+-----+---+-----+
         | 39 |  840   |DFI |ORG|Rsvd | RD |Area |ID |Sel |
         +----+--------+----+---+-----+----+-----+---+-----+
  octets | 1  |   2    | 1  | 3 |  2  | 2  |  2  | 6 | 1  |
         +----+--------+----+---+-----+----+-----+---+-----+
```

Figure 4-2 (b): ANSI NSAP address format for DCC=840

Definitions:

IDP	Initial Domain Part
AFI	Authority and Format Identifier
IDI	Initial Domain Identifier
DSP	Domain Specific Part
DFI	DSP Format Identifier
AA	Administration Authority
ORG	Organization Name (numeric form)
Rsvd	Reserved
RD	Routing Domain Identifier
Area	Area Identifier
ID	System Identifier
Sel	NSAP Selector

4.9.1 Destination Address Length Indicator

This field indicates the length, in octets, of the Destination Address.

4.9.2 Destination Address

This field contains an OSI NSAP address, as described in Section 4.9. It MUST always contain the address of the final destination. (This is true even for packets containing a source route option, see Section 4.13.4).

The final octet of the destination address MUST always contain the value of the PROTO field, as defined in IP. The 8-bit PROTO field indicates the next level protocol used in the data portion of the CLNP datagram. The values for various protocols are specified in "Assigned Numbers" [15]. For the PROTO field, the value of zero (0) is reserved.

TUBA implementations that support TCP/UDP as well as OSI MUST use the protocol value (1Dh, Internet decimal 29) reserved for ISO transport protocol class 4.

4.9.3 Source Address Length Indicator

This field indicates the length, in octets, of the Source Address.

4.9.4 Source Address

This field contains an OSI NSAP address, as described in Section 4.9.

The final octet of the source address is reserved. It MAY be set to the protocol field value on transmission, and shall be ignored on reception (the value of zero MUST not be used).

4.10 Data Unit Identifier

Like the Identification field of IP, this 16-bit field is used to distinguish segments of the same (original) packet for the purposes of reassembly. This field is present when the fragmentation permitted flag is set to one.

4.11 Fragment Offset

Like the Fragment Offset of IP, this 16-bit is used to identify the relative octet position of the data in this fragment with respect to the start of the data submitted to CLNP; i.e., it indicates where in the original datagram this fragment belongs. The offset is measured in octets; the value of this field shall always be a multiple of eight (8). This field is present when the fragmentation permitted flag is set to one.

4.12 Total Length

The total length of the CLNP packet in octets is determined by the originator and placed in the Total Length field of the header. The Total Length field specifies the entire length of the original datagram, including both the header and data. This field MUST NOT be changed in any fragment of the original packet for the duration of the packet lifetime. This field is present when the fragmentation permitted flag is set to one.

4.13 Options

All CLNP options are "triplets" of the form <parameter code>, <parameter length>, and <parameter value>. Both the parameter code and length fields are always one octet long; the length parameter value, in octets, is indicated in the parameter length field. The following options are defined for CLNP for TUBA.

4.13.1 Security

The value of the parameter code field is binary 1100 0101. The length field MUST be set to the length of a Basic (and Extended) Security IP option(s) as identified in RFC 1108 [16], plus 1. Octet 1 of the security parameter value field -- the CLNP Security Format Code -- is set to a binary value 0100 0000, indicating that the remaining octets of the security field contain either the Basic or Basic and Extended Security options as identified in RFC 1108. This encoding points to the administration of the source address (e.g., ISOC) as the administration of the security option; it is thus distinguished from the globally unique format whose definition is reserved for OSI use. Implementations wishing to use a security option MUST examine the

PROTO field in the source address; if the value of PROTO indicates the CLNP client is TCP or UDP, the security option described in RFC 1108 is used.

[Note: If IP options change, TUBA implementations MUST follow the new recommendations. This RFC, or revisions thereof, must document the new recommendations to assure compatibility.]

The formats of the Security option, encoded as a CLNP option, is as follows. The CLNP option will be used to convey the Basic and Extended Security options as sub-options; i.e., the exact encoding of the Basic/Extended Security IP Option is carried in a single CLNP Security Option, with the length of the CLNP Security option reflecting the sum of the lengths of the Basic and Extended Security IP Option.

```
+--------+--------+--------+--------+--------+---//----+-
|11000100|XXXXXXXX|01000000|10000010|YYYYYYYY|         |      ...
+--------+--------+--------+--------+--------+---//----+----
 CLNP       CLNP     CLNP     BASIC   BASIC    BASIC
 OPTION    OPTION   FORMAT  SECURITY  OPTION   OPTION
 TYPE      LENGTH    CODE    TYPE     LENGTH   VALUE
 (197)                       (130)

                       ---+------------+------------+----//-------+
                  ...     |  10000101  |  000LLLLL  |             |
                     -----+------------+------------+----//-------+
                             EXTENDED     EXTENDED    EXTENDED OPTION
                             OPTION       OPTION          VALUE
                            TYPE (133)    LENGTH
```

The syntax, semantics and processing of the Basic and Extended IP Security Options are defined in RFC 1108.

4.13.2 Type of Service

[Note: Early drafts recommended the use of IP Type of Service as specified in RFC 1349. There now appears to be a broad consensus that this encoding is insufficient, and there is renewed interest in exploring the utility of the "congestion experienced" flag available in the CLNP QOS Maintenance option. This RFC thus recommends the use of the QOS Maintenance option native to CLNP.]

The Quality of Service Maintenance option allows the originator of a CLNP datagram to convey information about the quality of service requested by the originating upper layer process. Routers MAY use this information as an aid in selecting a route when more than one route satisfying other routing criteria is available and the

available routes are know to differ with respect to the following qualities of service: ability to preserve sequence, transit delay, cost, residual error probability. Through this option, a router may also indicate that it is experiencing congestion.

The encoding of this option is as follows:

```
+-----------+-----------+----------+
| 1100 0011 | 0000 0001 | 110ABCDE |
+-----------+-----------+----------+
 CLNP QOS     OPTION      QOS FLAGS
 TYPE (195)   LENGTH
```

The value of the parameter code field MUST be set to a value of binary 1100 0011 (the CLNP Quality of Service Option Code point). The length field MUST be set to one (1).

Bits 8-6 MUST be set as indicated in the figure. The flags "ABCDE" are interpreted as follows:

```
A=1  choose path that maintains sequence over
     one that minimizes transit delay
A=0  choose path that minimizes transit delay over
     one that maintains sequence
B=1  congestion experienced
B=0  no congestion to report
C=1  choose path that minimizes transit delay over
     over low cost
C=0  choose low cost over path that
     minimizes transit delay
D=1  choose pathe with low residual error probability over
     one that minimizes transit delay
D=0  choose path that minimizes transit delay over
     one with low residual error probability
E=1  choose path with low residual error probability over
     low cost
E=0  choose path with low cost over one with low
     residual error probability
```

4.13.3 Padding

The padding field is used to lengthen the packet header to a convenient size. The parameter code field MUST be set to a value of binary 1100 1100. The value of the parameter length field is variable. The parameter value MAY contain any value; the contents of padding fields MUST be ignored by the receiver.

```
+----------+----------+-----------+
| 11001100 | LLLLLLLL | VVVV VVVV |
+----------+----------+-----------+
```

4.13.4 Source Routing

Like the strict source route option of IP, the Complete Source Route option of CLNP is used to specify the exact and entire route an internet datagram MUST take. Similarly, the Partial Source Route option of CLNP provides the equivalent of the loose source route option of IP; i.e., a means for the source of an internet datagram to supply (some) routing information to be used by gateways in forwarding the internet datagram towards its destination. The identifiers encoded in this option are network entity titles, which are semantically and syntactically the same as NSAPAs and which can be used to unambiguously identify a network entity in an intermediate system (router).

The parameter code for Source Routing is binary 1100 1000. The length of the source routing parameter value is variable.

The first octet of the parameter value is a type code, indicating Complete Source Routing (binary 0000 0001) or Partial Source Routing (binary 0000 0000). The second octet identifies the offset of the next network entity title to be processed in the list, relative to the start of the parameter (i.e., a value of 3 is used to identify the first address in the list). The offset value is modified by each router using a complete source route or by each listed router using a partial source route to point to the next NET.

The third octet begins the list of network entity titles. Only the NETs of intermediate systems are included in the list; the source and destination addresses shall not be included. The list consists of variable length network entity title entries; the first octet of each entry gives the length of the network entity title that comprises the remainder of the entry.

4.13.5 Record Route

Like the IP record route option, the Record route option of CLNP is used to trace the route a CLNP datagram takes. A recorded route consists of a list of network entity titles (see Source Routing). The list is constructed as the CLNP datagram is forwarded along a path towards its final destination. Only titles of intermediate systems (routers) that processed the datagram are included in the recorded route; the network entity title of the originator of the datagram SHALL NOT be recorded in the list.

The parameter code for Record Route is binary 1100 1011. The length of the record route parameter value is variable.

The first octet of the parameter value is a type code, indicating Complete Recording of Route (0000 0001) or Partial Recording of Route (0000 0000). When complete recording of route is selected, reassembly at intermediate systems MAY be performed only when all fragments of a given datagram followed the same route; partial recording of route eliminates or "loosens" this constraint.

The second octet identifies the offset where the next network entity title entry (see Source Routing) MAY be recorded (i.e., the end of the current list), relative to the start of the parameter. A value of 3 is used to identify the initial recording position. The process of recording a network entity title entry is as follows. A router adds the length of its network entity title entry to the value of record route offset and compares this new value to the record route list length indicator; if the value does not exceed the length of the list, entity title entry is recorded, and the offset value is incremented by the value of the length of the network entity title entry. Otherwise, the recording of route is terminated, and the router MUST not record its network entity title in the option. If recording of route has been terminated, this (second) octet has a value 255.

The third octet begins the list of network entity titles.

4.13.6 Timestamp

[Note: There is no timestamp option in edition 1 of ISO/IEC 8473, but the option has been proposed and submitted to ISO/IEC JTC1/SC6.]

The parameter code value 1110 1110 is used to identify the Timestamp option; the syntax and semantics of Timestamp are identical to that defined in IP.

The Timestamp Option is defined in STD 5, RFC 791. The CLNP parameter code 1110 1110 is used rather than the option type code 68 to identify the Timestamp option, and the parameter value conveys the option length. Octet 1 of the Timestamp parameter value shall be encoded as the pointer (octet 3 of IP Timestamp); octet 2 of the parameter value shall be encoded as the overflow/format octet (octet 4 of IP Timestamp); the remaining octets shall be used to encode the timestamp list. The size is fixed by the source, and cannot be changed to accommodate additional timestamp information.

```
        +--------+--------+--------+--------+
        |11101110| length | pointer|oflw|flg|
        +--------+--------+--------+--------+
        |         network entity title      |
        +--------+--------+--------+--------+
        |             timestamp             |
        +--------+--------+--------+--------+
        |                 .                 |
                          .
```

5. Error Reporting and Control Message Handling

CLNP and IP differ in the way in which errors are reported to hosts. In IP environments, the Internet Control Message Protocol (ICMP, [7]) is used to return (error) messages to hosts that originate packets that cannot be processed. ICMP messages are transmitted as user data in IP datagrams. Unreachable destinations, incorrectly composed IP datagram headers, IP datagram discards due to congestion, and lifetime/reassembly time exceeded are reported; the complete internet header that caused the error plus (at least) 8 octets of the segment contained in that IP datagram are returned to the sender as part of the ICMP error message. For certain errors, e.g., incorrectly composed IP datagram headers, the specific octet which caused the problem is identified.

In CLNP environments, an unique message type, the Error Report type, is used in the network layer protocol header to distinguish Error Reports from CLNP datagrams. CLNP Error Reports are generated on detection of the same types of errors as with ICMP. Like ICMP error messages, the complete CLNP header that caused the error is returned to the sender in the data portion of the Error Report. Implementations SHOULD return at least 8 octets of the datagram contained in the CLNP datagram to the sender of the original CLNP datagram. Here too, for certain errors, the specific octet which caused the problem is identified.

A summary of the contents of the CLNP Error Report, as it is proposed for use in TUBA environments, is illustrated in Figure 5-1:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |        ........Data Link Header........       | NLP ID        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Header Length  |     Version   | Lifetime (TTL)| 000 | Type=ER |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  TOTAL Length of Error Report |           Checksum            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Dest Addr Len |               Destination Address...          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Destination Address...                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Destination Address...                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Destination Address...                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Destination Address...                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | PROTO field   | Src  Addr Len |  Source  Address...           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Source Address...                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Source Address...                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Source Address...                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               ... Source Address...                           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |       ... Source Address      | Reason for Discard (type/len) |
 |                               |   1100 0001   | 0000 0010     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Reason for Discard        |    Options...                 |
 |   code        |   pointer     |                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                           Options                             |
 :                                                               :
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                            Data                               |
 :                                                               :
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Note that each tick mark represents one bit position.

Figure 5-1. Error Report Format

5.1 Rules for processing an Error Report

The following is a summary of the rules for processing an Error Report:

* An Error Report is not generated to report a problem encountered while processing an Error Report.

* Error Reports MAY NOT be fragmented (hence, the fragmentation part is absent).

* The Reason for Discard Code field is populated with one of the values from Table 5-1.

* The Pointer field is populated with number of the first octet of the field that caused the Error Report to be generated. If it is not possible to identify the offending octet, this field MUST be zeroed.

* If the Priority or Type of Service option is present in the errored datagram, the Error Report MUST specify the same option, using the value specified in the original datagram.

* If the Security option is present in the errored datagram, the Error Report MUST specify the same option, using the value specified in the original datagram; if the Security option is not supported by the intermediate system, no Error Report is to be generated (i.e., "silently discard" the received datagram).

* If the Complete Source Route option is specified in the errored datagram, the Error Report MUST compose a reverse of that route, and return the datagram along the same path.

5.2 Comparison of ICMP and CLNP Error Messages

Table 5-1 provides a loose comparison of ICMP message types and codes to CLNP Error Type Codes (values in Internet decimal):

CLNP Error Type	Codes	ICMP Message	(Type, Code)
Reason not specified	(0)	Parameter Problem	(12, 0)
Protocol Procedure Error	(1)	Parameter Problem	(12, 0)
Incorrect Checksum	(2)	Parameter Problem	(12, 0)
PDU Discarded--Congestion	(3)	Source Quench	(4, 0)
Header Syntax Error	(4)	Parameter problem	(12, 0)
Need to Fragment could not	(5)	Frag needed, DF set	(3, 4)
Incomplete PDU received	(6)	Parameter Problem	(12, 0)
Duplicate Option	(7)	Parameter Problem	(12, 0)
Destination Unreachable	(128)	Dest Unreachable,Net unknown	(3, 0)
Destination Unknown	(129)	Dest Unreachable,host unknown	(3, 1)
Source Routing Error	(144)	Source Route failed	(3, 5)
Source Route Syntax Error	(145)	Source Route failed	(3, 5)
Unknown Address in Src Route	(146)	Source Route failed	(3, 5)
Path not acceptable	(147)	Source Route failed	(3, 5)
Lifetime expired	(160)	TTL exceeded	(11, 0)
Reassembly Lifetime Expired	(161)	Reassembly time exceeded	(11, 1)
Unsupported Option	(176)	Parameter Problem	(12, 0)
Unsupported Protocol Version	(177)	Parameter problem	(12, 0)
Unsupported Security Option	(178)	Parameter problem	(12, 0)
Unsupported Src Rte Option	(179)	Parameter problem	(12, 0)
Unsupported Rcrd Rte	(180)	Parameter problem	(12, 0)
Reassembly interference	(192)	Reassembly time exceeded	(11, 1)

Table 5-1. Comparison of CLNP Error Reports to ICMP Error Messages

Note 1: The current accepted practice for IP is that source quench should not be used; if it is used, implementations MUST not return a source quench packet for every relevant packet. TUBA/CLNP implementations are encouraged to adhere to these guidelines.

Note 2: There are no corresponding CLNP Error Report Codes for the following ICMP error message types:
- Protocol Unreachable (3, 2)
- Port Unreachable (3, 3)

[Note: Additional error code points available in the ER type code block can be used to identify these message types.]

6. Pseudo-Header Considerations

A checksum is computed on UDP and TCP segments to verify the integrity of the UDP/TCP segment. To further verify that the UDP/TCP segment has arrived at its correct destination, a pseudo-header consisting of information used in the delivery of the UDP/TCP segment is composed and included in the checksum computation.

To compute the checksum on a UDP or TCP segment prior to transmission, implementations MUST compose a pseudo-header to the UDP/TCP segment consisting of the following information that will be used when composing the CLNP datagram:

* Destination Address Length Indicator

* Destination Address (including PROTO field)

* Source Address Length Indicator

* Source Address (including Reserved field)

* A two-octet encoding of the Protocol value

* TCP/UDP segment length

If the length of the {source address length field + source address + destination address field + destination address } is not an integral number of octets, a trailing 0x00 nibble is padded. If GOSIP compliant NSAP addresses are used, this never happens (this is known as the Farinacci uncertainty principle). The last byte in the Destination Address has the value 0x06 for TCP and 0x11 for UDP, and the Protocol field is encoded 0x0006 for TCP and 0x0011 for UDP. If needed, an octet of zero is added to the end of the UDP/TCP segment to pad the datagram to a length that is a multiple of 16 bits.

[Note: the pseudoheader is encoded in this manner to expedite processing, as it allows implementations to grab a contiguous stream of octets beginning at the destination address length indicator and terminating at the final octet of the source address; the PROTOCOL field is present to have a consistent representation across IPv4 and CLNP/TUBA implementations.]

Figure 6-1 illustrates the resulting pseudo-header when both source and destination addresses are maximum length.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Dest Addr Len |               Destination Address...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Destination Address...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    (PROTO)    | Src  Addr Len |  Source  Address...           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               ... Source Address...                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| ...           | (Reserved)    |    Protocol                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   UDP/TCP segment length      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6-1. Pseudo-header

7. Security Considerations

ISO CLNP is an unreliable network datagram protocol, and is subject to the same security considerations as Internet Protocol ([5], [8]); methods for conveying the same security handling information recommended for IP are described in Section 4.13.1, Security Option.

8. Author's Address

David M. Piscitello
Core Competence
1620 Tuckerstown Road
Dresher, PA 19025

Phone: 215-830-0692
EMail: wk04464@worldlink.com

9. References

[1] ISO/IEC 8473-1992. International Standards Organization -- Data Communications -- Protocol for Providing the Connectionless Network Service, Edition 2.

[2] Callon, R., "TCP/UDP over Bigger Addresses (TUBA)", RFC 1347, Internet Architecture Board, May 1992.

[3] Postel, J., "Transmission Control Protocol (TCP)", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[4] Postel, J., "User Datagram Protocol (UDP)", STD 6, RFC 768, USC/Information Sciences Institute, September 1981.

[5] Postel, J., "Internet Protocol (IP)", STD 5, RFC 791, USC/Information Sciences Institute, September 1981.

[6] Chapin, L., "ISO DIS 8473, Protocol for Providing the Connectionless Network Service", RFC 994, March 1986.

[7] Postel, J., "Internet Control Message Protocol (ICMP)", STD 5, RFC 792, USC/Information Sciences Institute, September 1981.

[8] Braden, R., Editor, "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, Internet Engineering Task Force, October 1989.

[9] Hagens, R., "An Echo Function for ISO 8473", RFC 1139, IETF-OSI Working Group, May 1993.

[10] Sklower, K., "Improving the Efficiency of the ISO Checksum Calculation" ACM SIGCOMM CCR 18, no. 5 (October 1989):32-43.

[11] ISO/IEC 8348-1992. International Standards Organization--Data Communications--OSI Network Layer Service and Addressing.

[12] Callon, R., Gardner, E., and R. Hagens, "Guidelines for OSI NSAP Allocation in the Internet", RFC 1237, NIST, Mitre, DEC, July 1991.

[13] Piscitello, D., "Assignment of System Identifiers for TUBA/CLNP Hosts", RFC 1526, Bellcore, September 1993.

[14] ISO/IEC 9542:1988/PDAM 1. Information Processing Systems -- Data Communications -- ES/IS Routeing Protocol for use with ISO CLNP -- Amendment 1: Dynamic Discovery of OSI NSAP Addresses by End Systems.

[15] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1340 USC/Information Sciences Institute, July 1992.

[16] Kent, S., "Security Option for IP", RFC 1108, BBN Communications, November 1991.

Appendix A. Checksum Algorithms (from ISO/IEC 8473)

Symbols used in algorithms:

c0, c1	variables used in the algorithms
i	position of octet in header (first octet is i=1)
Bi	value of octet i in the header
n	position of first octet of checksum (n=8)
L	Length of header in octets
X	Value of octet one of the checksum parameter
Y	Value of octet two of the checksum parameter

Addition is performed in one of the two following modes:

* modulo 255 arithmetic;

* eight-bit one's complement arithmetic;

The algorithm for Generating the Checksum Parameter Value is as follows:

A. Construct the complete header with the value of the checksum parameter field set to zero; i.e., c0 <- c1 <- 0;

B. Process each octet of the header sequentially from i=1 to L by:

   * c0 <- c0 + Bi

   * c1 <- c1 + c0

C. Calculate X, Y as follows:

   * X <- (L - 8)(c0 - c1) modulo 255

   * Y <- (L - 7)(-C0) + c1

D. If X = 0, then X <- 255

E. If Y = 0, then Y <- 255

F. place the values of X and Y in octets 8 and 9 of the header, respectively

The algorithm for checking the value of the checksum parameter is as follows:

A. If octets 8 and 9 of the header both contain zero, then the checksum calculation has succeeded; else if either but not both of these octets contains the value zero then the checksum is incorrect; otherwise, initialize: c0 <- c1 <- 0

B. Process each octet of the header sequentially from i = 1 to L by:

   * c0 <- c0 + Bi

   * c1 <- c1 + c0

C. When all the octets have been processed, if c0 = c1 = 0, then the checksum calculation has succeeded, else it has failed.

There is a separate algorithm to adjust the checksum parameter value when a octet has been modified (such as the TTL). Suppose the value in octet k is changed by Z = newvalue - oldvalue. If X and Y denote the checksum values held in octets n and n+1 respectively, then adjust X and Y as follows:

If X = 0 and Y = 0 then do nothing, else if X = 0 or Y = 0 then the checksum is incorrect, else:

```
X <- (k - n - 1)Z + X   modulo 255

Y <- (n - k)Z + Y       modulo 255
```

If X = 0, then X <- 255; if Y = 0, then Y <- 255.

In the example, n = 89; if the octet altered is the TTL (octet 4), then k = 4. For the case where the lifetime is decreased by one unit (Z = -1), the assignment statements for the new values of X and Y in the immediately preceeding algorithm simplify to:

```
X <- X + 5      Modulo 255

Y <- Y - 4      Modulo 255
```